ClearEx Compiler

Objective:

For better understanding of the backend processes of the simple compiler and to know the basic function including what tokens, regular expressions have been used in this specific Project/Lexical Analyzer.

Abstract:

The need that meets the user is the basic function to note down and collect the tokens that have been defined in the writing of the program and in the end printing down the contents of the written file. Each form of data has been separately classified as per its nature. This Lexical Analyzer/Compiler has its own specific expressions that would be better for someone having the same sort of need. This can improve one's efficiency and speed and also help one to better understand the backend workflow.

Lexical Units:

\n	New line
\t	Tab
\\	Backslash

Symbols: ++, +=, =, -=, --

Expressions:

lvalue := expr

id

if expr else expr

if expr else if expr else expr

Grammar Rules:

PRINT	print
[A-Z]	IDENTIFIER,
[a-z]	STRING
","	COMMA
"["	LEFT_BRACKET
"]"	RIGHT_BRACKET
'='	ASSIGNMENT
"∪"	UNION
"∩"	INTERSECT
"-"	SUBSTRACT
\n	NEWLINE
\"//".*	comment
[\t]	space or tab
.	unexpected token

{increment_operators}
{decrement_operators}

Sample Program:

```
%option noyywrap
```

```
%{

   #include <stdio.h>


%}


increment_operators (\++|\+=)

decrement_operators (\-=|\--)


%%


PRINT       { printf("PRINT, '%s'\n",
yytext); }

[A-Z]       { printf("IDENTIFIER,
'%s'\n", yytext); }

[a-z]       { printf("STRING, '%s'\n",
yytext); }

","         { printf("COMMA, '%s'\n",
yytext); }

"["         { printf("LEFT_BRACKET,
'%s'\n", yytext); }

"]"         {
printf("RIGHT_BRACKET, '%s'\n",
yytext); }

"="         { printf("ASSIGNMENT,
'%s'\n", yytext); }

"∪"         { printf("UNION, '%s'\n",
yytext); }

"∩"         { printf("INTERSECT,
'%s'\n", yytext); }

"-"         { printf("SUBSTRACT,
'%s'\n", yytext); }

\n          { printf("NEWLINE\n"); }

"//".*      { printf("comment: %s\n",
yytext); }

[ \t]       { printf("space or tab\n"); }

.           { printf("unexpected token:
(%s)\n", yytext); }

{increment_operators}
printf("(increment_operator) '%s'
",yytext);

{decrement_operators}
printf("(decrement_operator) '%s'
",yytext);


%%


int main()

{


        printf("Hello World");


   yylex();

}
```